
Introducción a Ruby

— Control de Acceso —

Overview

- Los tres niveles de control de acceso.
- Cómo controlamos el acceso?
- Qué tan privado es el acceso privado?

Control de acceso

- Cuando **diseñamos** una clase, es **importante** pensar en **qué tanto** de lo que tenemos **queremos exponer** al mundo.
- **Encapsulación**: intenta esconder la **representación interna** de un objeto para que podamos **cambiarlo** luego.
- Los tres niveles de control de acceso son: **public, protected y private**
-
-


Encapsulación

```
class Car
  def initialize(speed, comfort)
    @rating = speed * comfort
  end

  # Can't SET rating from outside
  def rating
    @rating
  end
end

puts Car.new(4, 5).rating # => 20
```

Los detalles de cómo calculamos rating son internos a la clase.... Nadie puede setear un rating desde afuera...



Especificando el control de acceso

- Existen **dos** formas de especificar el control de acceso:
 - **Especificar `public`, `protected` o `private`.**
 - Todo lo que esté debajo de este token será de ese nivel de control de acceso.
 - **Definir** los métodos regularmente y después especificar cuáles son `public`, `protected` o `private` y **listarlos** separados por coma utilizando los **símbolos de métodos**.

Especificando el control de acceso

```
class MyAlgorithm
  private
    def test1
      "Private"
    end
  protected
    def test2
      "Protected"
    end
  public
    def public_again
      "Public"
    end
end
```

```
class Another
  def test1
    "Private, as declated later on"
  end
  private :test1
end
```

Control de acceso

- Los métodos **public** - **no** tienen ningún control de acceso
 - **Cualquiera** puede invocar estos métodos.
- Los métodos **protected** - pueden ser invocados por los objetos de la **clase** o sus **subclases**.
- Los objetos **private** - **no** pueden ser invocados con un receiver explícito:
 - **Existe una excepción** a esta regla: los métodos “setter” pueden ser invocados con un receiver explícito.

Acceso Privado

```
class Person
  def initialize(age)
    self.age = age # LEGAL - EXCEPCION A LA REGLA
    puts my_age
    # puts self.my_age # ILEGAL
    # NO SE PUEDE USAR self o cualquier otro receiver
  end

  private
  def my_age
    @age
  end
  def age=(age)
    @age = age
  end
end
```

```
Person.new(25) # => 25
```


Entonces

- Los modificadores `public` y `private` son los controles de acceso mas utilizados.
- Los métodos privados no son invocables desde afuera o desde adentro de la clase con un receptor(receiver) explícito.